# 如何模拟抽卡概率?

## 2021/8/30 Leetcode.528 按权重随机选择

- 题目:

```
给定一个正整数数组 w ，其中 w[i] 代表下标 i 的权重（下标从 0 开始），请写一个函数 pickIndex ，它可以随机地获取下标 i，选取下标 i 的概率与 w[i] 成正比。

例如，对于 w = [1, 3]，挑选下标 0 的概率为 1 / (1 + 3) = 0.25 （即，25%），而选取下标 1 的概率为 3 / (1 + 3) = 0.75（即，75%）。

也就是说，选取下标 i 的概率为 w[i] / sum(w) 。
```

- 示例:

```
输入：
["Solution","pickIndex","pickIndex","pickIndex","pickIndex","pickIndex"]
[[[1,3]],[],[],[],[],[]]
输出：
[null,1,1,1,1,0]
解释：
Solution solution = new Solution([1, 3]);
solution.pickIndex(); // 返回 1，返回下标 1，返回该下标概率为 3/4 。
solution.pickIndex(); // 返回 1
solution.pickIndex(); // 返回 1
solution.pickIndex(); // 返回 1
solution.pickIndex(); // 返回 0，返回下标 0，返回该下标概率为 1/4 。

由于这是一个随机问题，允许多个答案，因此下列输出都可以被认为是正确的:
[null,1,1,1,1,0]
[null,1,1,1,1,1]
[null,1,1,1,0,0]
[null,1,1,1,0,1]
[null,1,0,1,0,0]
......
诸若此类。
```

- 思路:

这应该是我目前为止见过最实用的算法题了，什么手游抽卡模拟器啊嗯🤔？言归正传，这道题可能有很多小伙伴和我一样刚看到题感觉很熟悉但又无从下手。冷静下来想了想，这不就是初中高中几个红球黑球黄球混在一起求抽到某个球的概率吗？虽然我们初中高中都是给出几个球求概率，而这里是给出了球，然后默认计算了概率，最后让你模拟抽球。所以我们感到陌生的就是这里，求概率当然简单，加了一个模拟抽球就无从下手了🤔。我们得回归题目本质，我们终归是抽球，不要在意抽的概率——对，我们是要在众多球里抽出一个球。我们要先把所有下标看成各种不同颜色的球，然后把w[i]看成这个球出现多少次并一一标号，再把它们丢到黑盒里抽。反正每次只抽一个，故交给rand()函数即可。

由于上面我们是从1开始一一标号，故整个球的序列是从1到total（total是权重和）。然后开始分区：例如w[3,1,2,4]，我们分区得到[1,2,3]，[4]，[5,6]，[7,8,9,10]四个区，每个区对应每个下标的小球，这样[1,2,3.....,10]就模拟出抽球的环境了。注意到，rand()每次只能抽一个数，所以我们得到的1到total里的数要映射出对应的球。

怎么映射呢？直接开一个数组摆上1到total吗？显然不行，当数字很大时，明显花费很大的空间！其实整个分区是抽象出来的，我们要的只是一个映射关系，例如，我抽到了6，我知道它大于4，小于等于6，这关系就指出了它位于第二个和第四个区间中间。所以我们用到了前缀和模拟区间的上分界：[3,4,6,10]，如此，抽到5，知道它大于4，小于6，故位于第三区间；抽到1，它没大的，但不妨碍它小于3，归到1区间。

这样一来，我们可以通过抽到一个数，从头挨个比较它和各前缀和的大小，直到它小于等于某个前缀和便停止，对应的前缀和下标即区间，就是所求答案。由于有严格的递增关系，故可用二分查找加速查找过程。

- 代码：

这是力扣官方代码，写的足够好，无需我这个垃圾魔改😏

```
typedef struct {
    int* pre;       //前缀和数组
    int preSize;    //前缀和数组的长度
    int total;      //权重总和
} Solution;         //一个Solution代表一组w[]

//对这组w[]进行预处理，求出前缀和
Solution* solutionCreate(int* w, int wSize) {
    Solution* obj = malloc(sizeof(Solution));
    obj->pre = malloc(sizeof(int) * wSize);
    obj->preSize = wSize;
    obj->total = 0;
    for (int i = 0; i < wSize; i++) {
        obj->total += w[i];
        if (i > 0) {
            obj->pre[i] = obj->pre[i - 1] + w[i];
        } else {
            obj->pre[i] = w[i];
        }
    }
    return obj;
}

//用二分查找抽到的球根据对应的前缀和求出落在的区间
int binarySearch(Solution* obj, int x) {
    int low = 0, high = obj->preSize - 1;
    while (low < high) {
        int mid = (high - low) / 2 + low;
        if (obj->pre[mid] < x) {
            low = mid + 1; //如果比前缀和上界大，那必是在其后面的区间
        } else {
            high = mid;    //否则，必是在包含该区间的前面众多区间中
        }
    }
    return low;
}

int solutionPickIndex(Solution* obj) {
    int x = rand() % obj->total + 1;   //先随机出一个数
    return binarySearch(obj, x);       //二分查找
}

void solutionFree(Solution* obj) {
```

```
    free(obj->pre);
    free(obj);
}
```

- 后记:

这个是真的实用啊🤣🤣🤣